CSC 282 - Fall 2009 - http://www.cs.rochester.edu/~stefanko/Teaching/09CS282/

QUIZ #4 - this quiz contributes 12% to your grade.

100 points total

1. (-24 to 26 points) For each statement below (try to) determine its truth value. Grading: correct answer 2 points, no answer 0 points, wrong answer -3 points, all answers correct $\Longrightarrow$ 10 points.

   - For each collection of coin values below determine whether the greedy algorithm always produces an optimal solution (YES = greedy always optimal, NO = greedy sometimes not optimal).

     (a) $1, 2, 5$ YES - NO

     (b) $1, 5, 11$ YES - NO

     (c) $1, 4, 13$ YES - NO

     (d) $1, 2, 4, 8, 16$ YES - NO

   - Recall the activity selection problem: An activity is a time interval [start-time, end-time]. Two activities are said to conflict if they overlap. The goal is to select the largest subset S of given activities such that no pair of (distinct) activities in S conflict.

     For each of the following 4 greedy algorithms decide if it always gives an optimal subset of the activities. Circle the correct answer (YES = always optimal, NO = sometimes not optimal).

     (a) Start with $S = \{\}$, $Z =$ INPUT. Repeat the following until no activity is left. Pick the activity $a \in Z$ with the earliest start-time, put a into S, and remove all activities in Z conflicting with a.

     YES - NO

     (b) Start with $S = \{\}$, $Z =$ INPUT. Repeat the following until no activity is left. Pick the activity $a \in Z$ with the earliest end-time, put a into S, and remove all activities in Z conflicting with a.

     YES - NO

     (c) Start with $S = \{\}$, $Z =$ INPUT. Repeat the following until no activity is left. Pick the activity $a \in Z$ with the latest start-time, put a into S, and remove all activities in Z conflicting with a.

     YES - NO

     (d) Start with $S = \{\}$, $Z =$ INPUT. Repeat the following until no activity is left. Pick the activity $a \in Z$ with the latest end-time, put a into S, and remove all activities in Z conflicting with a.

     YES - NO

2. (10 points) Let $\overline{x} = x_1, \ldots, x_m$, $\overline{y} = y_1, \ldots, y_n$, and $\overline{z} = z_1, \ldots, z_\ell$ be three sequences. Our goal is to compute the length of the longest common subsequence of $\overline{x}, \overline{y}$, and $\overline{z}$. We will have a 3-dimensional table $T[0..m, 0...n, 0..\ell]$, where $T[i, j, k]$ will be the length of the longest common subsequence of $x_1, \ldots, x_i$, $y_1, \ldots, y_j$, and $z_1, \ldots, z_k$. Give an expression (or a piece of code) to compute $T[i, j, k]$ from previously computed values in T.

3. (10 points) Let $m \leq n$ be positive integers. Let $P[1..n]$ be an array of positive real numbers. We are given a chocolate bar consisting of $m \times n$ squares. We would like to split the bar into 1×1 squares. We are only allowed to split one piece of the chocolate at a time using a vertical or a horizontal break. We have to pay $P[\ell]$ for a break of length ℓ. We would like to compute the minimal cost of breaking the bar.

   We want solve the problem using dynamic programming. Let $M[i, j]$ be the minimal cost of breaking an $i \times j$ chocolate bar. In the space below write an expression for $M[i, j]$ in terms of

   $$M[1, j], M[2, j], \ldots, M[i-1, j],\ M[i, 1], M[i, 2], \ldots, M[i, j-1], \text{ and } P[i], P[j].$$

   $M[i, j] =$

   (Example: a 2×3 bar can be broken as follows (using 2 breaks of length 2 and 3 breaks of length 1):

   $$2 \times 3 \to 2 \times 2, 2 \times 1 \to 2 \times 1, 2 \times 1, 2 \times 1 \to \ldots \to 6 \text{ copies of } 1 \times 1$$

   It can also be broken as follows (using 1 break of length 3 and 4 breaks of length 1):

   $$2 \times 3 \to 1 \times 3, 1 \times 3 \to \ldots \to 6 \text{ copies of } 1 \times 1.$$

   in this case the answer would be the smaller of $2P[2]+3P[1]$ and $P[3]+4P[1]$ (say, if, $P[1] = 1, P[2] = 2, P[3] = 10$ then the minimal cost is 7)).

4. (10 points) Find frequencies for symbols a, b, c, d such that the the following table gives Huffman encoding (put the frequencies in the third column).

a	0	
b	10	
c	110	
d	111	

5. (10 points) Find the longest increasing subsequence of the sequence below (circle the elements of the subsequence). Under each of the numbers write the length of the longest increasing subsequence ending with that number.

8, 10, 8, 4, 14, 7, 9, 1, 2, 11, 13

6. (10 points) Consider the coin-change problem with coin values $1, 2, 6, 13$. Give an amount for which the greedy algorithm does not use the minimal number of coins.

7. (10 points) Given a string $\overline{x} = x_1 x_2 \cdots x_n$ we would like to find the length of the longest palindromic subsequence of $\overline{x}$ (a sequence is palindromic if it is the same as its reverse). Let $T[1..n, 1..n]$ be an array where $T[i, j]$ is the length of the longest palindromic subsequence of $x_i, \ldots, x_j$ (note that, $T[i, j]$ is undefined for $j < i$). Give an expression (or a piece of code) for $T[i, j]$ in terms of already computed values in T.

8. (10 points) We are given a collection of n intervals $I_1, \ldots, I_n$. Interval $I_i = [a_i, b_i]$ is assigned weight w_i. We want to find a max-weight subset of disjoint intervals, i. e., we want to find $S \subseteq \{1, \ldots, n\}$ such that

   - for any distinct $j, k \in S$ the intervals I_j and I_k are disjoint, and
   - the $\sum_{j \in S} w_j$ is maximized.

   Give an $O(n^2)$-time dynamic programming algorithm for this problem (clearly state what the content of the table is and explain the update rule).

   (To simplify the exposition you can assume that the a_i and b_i are all distinct.)

9. (10 points) In the KNAPSACK PROBLEM we have n items. The weight of the i-th item is $W[i]$ and the value of the i-th item is $V[i]$. Assume that the $V[i]$ are integers and $W[i]$ are real numbers. Let C be the capacity of the knapsack, and let M be the sum of the $V[i]$, that is, $M = \sum_{i=1}^{n} V[i]$. We would like to find a subset of items $S \subseteq \{1, \ldots, n\}$ with total weight at most C and maximal value.

   We will compute an array $K[0..M, 0..n]$, where entry $K[x, i]$ will be the minimal weight of a subset of $\{1, \ldots, i\}$ with total value equal to x (if no subset of $\{1, \ldots, i\}$ has total value x then the entry will be ∞). Give an expression (or a piece of code, if you wish) for $K[v, i]$ in terms of some of the $K[?, i-1]$ (the question mark should be replaced by appropriate expressions).

   $K[v, i] =$

10. (10 points) We are given an $n \times n$ array A of zeros and ones. We want to find the size of the largest contiguous all-ones square. We are going to give a dynamic-programming algorithm with running time $O(n^2)$.

    We will compute an $n \times n$ array P, where $P[i, j]$ is the size of the largest contiguous all-ones square whose bottom-right corner is i, j. We have $P[i, j] = A[i, j]$ if $i = 1$ or $j = 1$. If $i > 1$ and $j > 1$ then we will compute $P[i, j]$ using the values of $P[i-1, j], P[i-1, j-1], P[i, j-1]$, and $A[i, j]$. Give an expression (or a piece of code) for $P[i, j]$.

    $P[i, j] =$

11. (10 points) We are given n positive numbers $a_1, \ldots, a_n$ (the numbers are not necessarily integers). The goal is to select a subset of the numbers with maximal sum and such that no two consecutive numbers are selected. We are going to give an $O(n)$-time algorithm for the problem. Let $P[i]$ be the maximum sum of a subset of the first i numbers. Give an expression (or a piece of code) for $P[i]$ in terms of $P[i-2]$, $P[i-1]$, and a_i.

$$P[i] =$$

12. (10 bonus points) In the KNAPSACK PROBLEM REVISITED we have n items. The weight of the i-th item is $W[i]$ and the value of the i-th item is $V[i]$. Now we assume that the $W[i]$ are integers and $V[i]$ are real numbers. In contrast to the original knapsack problem we will assume that **the thief has 2 knapsacks** with capacities C_1 and C_2. We would like to find two disjoint subsets of items $S_1, S_2 \subseteq \{1, \ldots, n\}$ with maximal total value, and such that the total weight of S_1 is at most C_1 and the total weight of S_2 is at most C_2.

We will compute an array $K[0..C_1, 0...C_2, 0..n]$, where entry $K[x_1, x_2, i]$ will be the value of the solution restricted to items $\{1, \ldots, i\}$ with two knapsacks with capacities x_1 and x_2, formally,

$$K[x, y, i] = \max \left\{ \sum_{i \in S_1 \cup S_2} V[i] \;\middle|\; S_1, S_2 \subseteq \{1, \ldots, i\}, S_1 \cap S_2 = \emptyset, (\forall j \in \{1, 2\}) \sum_{i \in S_j} W[i] \leq x_j \right\}.$$

Give an expression (or a piece of code) for $K[x_1, x_2, i]$ in terms of $K[x_1, x_2, i-1]$, $K[x_1 - ?, x_2, i-1]$, $K[x_1, x_2 - ?, i-1]$, $V[i]$ and $W[i]$.

$$K[x, y, i] =$$